Particle Filter Networks with Application to Visual Localization

Peter Karkus[1,2] **David Hsu**[1,2] **Wee Sun Lee**[2]

[1]NUS Graduate School for Integrative Sciences and Engineering
[2]School of Computing
National University of Singapore
{karkus, dyhsu, leews}@comp.nus.edu.sg

Abstract: Particle filtering is a powerful approach to sequential state estimation and finds application in many domains, including robot localization, object tracking, etc. To apply particle filtering in practice, a critical challenge is to construct probabilistic system models, especially for systems with complex dynamics or rich sensory inputs such as camera images. This paper introduces the Particle Filter Network (PF-net), which encodes both a system *model* and a particle filter *algorithm* in a single neural network. The PF-net is fully differentiable and trained end-to-end from data. Instead of learning a generic system model, it learns a model optimized for the particle filter algorithm. We apply the PF-net to a visual localization task, in which a robot must localize in a rich 3-D world, using only a schematic 2-D floor map. In simulation experiments, PF-net consistently outperforms alternative learning architectures, as well as a traditional model-based method, under a variety of sensor inputs. Further, PF-net generalizes well to new, unseen environments.

Keywords: sequential state estimation, particle filtering, deep neural network, end-to-end learning, visual localization

1 Introduction

Particle filtering, also known as the sequential Monte-Carlo method, is a powerful approach to sequential state estimation [1]. Particle filters are used extensively in robotics, computer vision, physics, econometrics, etc. [2, 3, 4, 5, 6, 7], and are critical for robotic tasks such as localization [8], SLAM [9], and planning under partial observability [10]. To apply particle filters in practice, a major challenge is to construct probabilistic system models or learn them from data [11, 12, 13]. Consider, for example, robot localization with an onboard camera (Figure 1a). The observation model is a probability distribution over all possible camera images, conditioned on a continuous robot state and an environment map. Learning such a model is challenging, because of the enormous observation space and the lack of sufficient labeled data. An emerging line of research circumvents the difficulty of traditional model learning: it embeds an algorithm into a deep neural network and then performs end-to-end learning to train a model optimized for the specific algorithm [14, 17, 18, 19, 20].

In this direction, we introduce the Particle Filter Network (PF-net), a recurrent neural network (RNN) with differentiable algorithm prior for sequential state estimation. A PF-net encodes learnable probabilistic state-transition and observation *models* together with the particle filter *algorithm* in a single neural network (Figure 1b). It is fully differentiable and trained end-to-end from data. PF-net tackles the key challenges of learning complex probabilistic system models. Neural networks are capable of representing complex models over large spaces, e.g., observation models over images. Further, the network representation unites the model and the algorithm and thus allows training end-to-end. As a result, PF-net learns system models optimized for a specific algorithm, in this case, particle filtering, instead of learning generic system models. The models may learn only the features relevant for state estimation, thus reducing the complexity of learning.

We apply PF-net to robot visual localization, which is of great interest to mobile robotics. A robot navigates in a previously unseen environment and does not know its own precise location. It must

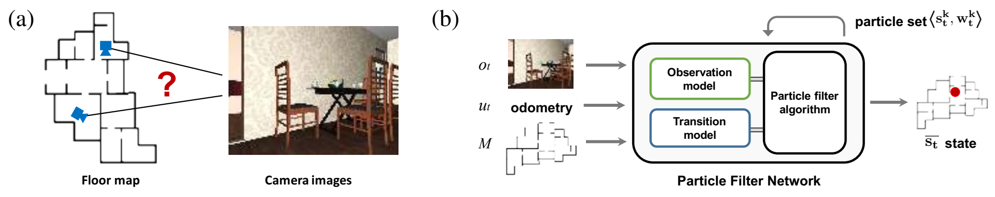

Figure 1: (a) Robot visual localization in a 3-D environment, using only a schematic 2-D floor map. The robot must match rich 3-D visual features with crude 2-D geometric features from the map. It must also ignore objects not in the map, e.g., furniture. (b) PF-net encodes both a learned probabilistic system model and the particle filter algorithm in a single neural network. It trains the model end-to-end in the context of the particle filter algorithm, resulting in improved performance.

localize in a visually rich 3-D world, given only a schematic 2-D floor map and observations from onboard sensors (Figure 1a). While particle filtering is the standard approach for LIDAR [8], we consider visual sensors, e.g., cameras. Now the probabilistic observation model must match rich 3-D visual features from camera images to crude 2-D geometric features from the map. Further, the camera images may contain various objects not in the map, e.g., furniture. This task exhibits key difficulties of state estimation from ambiguous, partial observations. A standard model-based approach would construct an observation model as a probability distribution of images conditioned on the floor map and robot pose. This is difficult, because of the enormous observation space, i.e., the space of all possible images showing various floor layouts, furniture configurations, etc. In contrast, PF-net trains a model end-to-end and learns only features relevant to the localization task.

This paper makes two contributions. First, we encode a particle filter algorithm in a neural network to learn models for sequential state estimation end-to-end. Second, we apply PF-net to visual localization and present a network architecture for matching rich visual features of a 3-D world with a schematic 2-D floor map. Simulation experiments on the House3D data set [22] show that the learned PF-net is effective for visual localization in new, unseen environments populated with furniture. Through end-to-end training, it also outperforms a conventional model-based method; it fuses information from multiple sensors, in particular, RGB and depth cameras; and it naturally integrates semantic information for localization, such as map labels for doors and room types.

2 Background

2.1 Related work

The idea of differentiable algorithm priors, i.e., embedding algorithms into a deep neural network, has been gaining attention recently. It has led to promising results for graph search [18, 19, 23], path integral optimal control [24], quadratic optimization [20, 25], and decision-making in fully observable environments [14] and partially observable environments [17, 21].

The general idea, when applied to probabilistic state estimation, has led to, e.g., Kalman filter network [15] and histogram filter network [16]. However, Kalman filtering assumes that the underlying state distribution is or can be well approximated as a unimodal Gaussian. Histogram filtering assumes discrete state spaces and has difficulty in scaling up to high-dimensional state spaces because of the "curse of dimensionality". To tackle arbitrary distributions and very large discrete or continuous state spaces, one possibility is particle filtering. Concurrent to our work, Jonschkowski et al. have been independently working on the idea of differentiable particle filtering [26]. The work is closely related, and we want to highlight several important differences. First, we propose a differentiable approximation of resampling, a crucial step for many particle filter algorithms. Next, we apply PF-net to visual localization in *new, unseen* environments, after learning. While the concurrent work also deals with localization, it does so in a fixed environment. Finally, our observation model for visual localization matches rich 3-D visual feature with a schematic 2-D floor map, ignores objects not in the map, and fuses information from multiple sources. Neural networks have been used with particle filters in variational learning as well. Unlike the PF-net, such networks aim to parameterize a family of generative distributions over observations [27, 28, 29, 30], thus making them unsuitable for large, complex observation spaces, such as the space of camera images and floor maps.

Particle filter methods, e.g., Monte-Carlo localization [8], are standard solutions to mobile robot localization. Many such methods assume a LIDAR sensor mounted on the robot and rely on handcrafted simple analytic observation models [31]. While there have been attempts to incorporate monocular or depth cameras [32, 33, 34, 35], constructing probabilistic observation models for them remains a challenge. PF-net learns effective system models through end-to-end training, without direct supervision on model components.

2.2 Particle filter algorithm

Particle filters periodically approximate the posterior distribution over states after an observation is received, i.e., they maintain a belief over states, $b(s)$. The belief is approximated by a set of *particles*, i.e., weighted samples from the probability distribution,

$$b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}, \tag{1}$$

where $\sum_k w_k = 1$, K is the number of particles, s_k is the particle state, w_k is the particle weight, and t denotes time. Importantly, the particle set can approximate arbitrary distributions, e.g., continuous, multimodal, non-Gaussian distributions. The state estimate can be computed by the weighted mean, $\overline{s}_t = \sum_k w_t^k s_t^k$. The particles are periodically updated in a Bayesian manner. First, the particle states are updated by sampling from a probabilistic transition model,

$$s_t^k \sim T(s_t | u_t, s_{t-1}^k), \tag{2}$$

where the transition model, T, defines the probability of a state, s_t, given a previous state, s_{t-1}^k, and the last action, u_t. In the case of robot localization u_t is the odometry input. Second, the particle weights are updated. The likelihood, f_t^k, is computed for each particle,

$$f_t^k = Z(o_t | s_t^k; \mathbb{M}), \tag{3}$$

where the observation model, Z, defines the conditional probability of an observation, o_t, given a state and the 2-D floor map, $\mathbb{M}$. Particle weights are updated according to the likelihoods,

$$w_t^k = \eta f_t^k w_{t-1}^k, \tag{4}$$

where $\eta^{-1} = \sum_{j=1:K} f_t^j w_{t-1}^j$ is a normalization factor.

One common issue is particle degeneracy, i.e., when most particles have near-zero weight. The issue can be addressed by *resampling* particles. New particles are sampled from the current set with repetition, where a particle is chosen with a probability proportionate to its weight,

$$p(k) = w_t^k. \tag{5}$$

The weights are updated according to a uniform distribution,

$$w_t'^k = 1/K. \tag{6}$$

The new particle set approximates the same distribution, but devotes its representation power to the important regions of the belief space. Note that the new set may contain repeated particles, but they diverge after stochastic transition updates.

3 Particle Filter Network

The Particle Filter Network (PF-net) encodes learnable transition and observation models, together with the particle filter algorithm, in a single neural network (Figure 1b). PF-net is a RNN with differentiable algorithm prior, that is, structure specific to sequential state estimation. The differentiable algorithm prior in PF-net is particle filtering: the particle representation of beliefs, and Bayesian updates for transitions and observations. Compared to generic architectures, such as LSTM [36], these priors allow much more efficient learning.

The key idea underlying our approach is the unified representation of a learned model and an inference algorithm. The model is a neural network, i.e., a computation graph with trainable parameters. The inference algorithm is a differentiable program, i.e., a computation graph with differentiable operations. Both the model and the algorithm can be encoded in the same computation graph. What

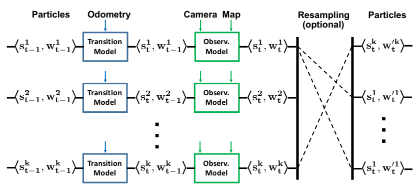

Figure 2: PF-net as a computation graph. The state-transition and observation models are captured in network weights, which are shared across the particles.

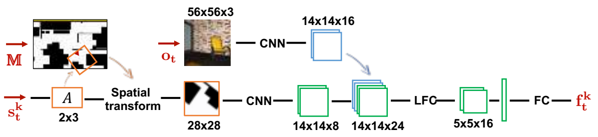

Figure 3: The PF-net observation model. The inputs are floor map $\mathbb{M}$, observation o_t, and particle state s_t^k. The output is particle likelihood f_t^k. CNN, LFC and FC are convolutional, locally fully-connected, and fully connected network components, respectively.

is the benefit? Unlike conventional model learning methods, PF-net can learn a model end-to-end, backpropagating gradients through the inference algorithm. The model is now optimized for a specific inference algorithm and a specific task. As a result, the model may not need to capture complex conditional probability distributions, instead, it may learn only the features relevant to the task.

Specifically, PF-net encodes the particle filtering steps, (1)-(6), in a computation graph (Figure 2). The transition and observation models, (2) and (3), are trainable neural networks with appropriate structure. Learned network weights are shared across particles. The rest of the computation graph is not learned, but rather, it implements the operations (1)-(6). Importantly, these operations must be differentiable to allow backpropagation. This is an issue for sampling from a learned distribution in (2), and resampling particles in (5)-(6).

The sampling operation (2) is not differentiable, but it can be easily expressed in a differentiable manner using the "reparameterization trick" [37, 38]. The trick is to take a noise vector as input, and express the desired distribution as a deterministic, differentiable function of this input. The function may have learnable parameters, e.g., the mean and variance of a Gaussian. Particle resampling poses a different issue: new particle weights are set to constant in (5), which produces zero gradients. We address the issue by introducing *soft-resampling*, a differentiable approximation based on importance sampling. Instead of sampling particles from the desired distribution $p(k)$, we sample from $q(k)$, a combination of $p(k)$ and a uniform distribution,

$$q(k) = \alpha w_t^k + (1 - \alpha)1/K, \tag{7}$$

where α is a trade-off parameter. The new weights are computed by the importance sampling formula,

$$w_t'^k = \frac{p(k)}{q(k)} = \frac{w_t^k}{\alpha w_t^k + (1-\alpha)1/K}. \tag{8}$$

This operation has non-zero gradient when $\alpha \neq 1$. Soft-resampling trades off the desired sampling distribution ($\alpha = 1$) with the uniform sampling distribution ($\alpha = 0$). It provides non-zero gradients by maintaining the dependency on previous particle weights. An alternative to soft-resampling is to simply carry over particles to the next step, without resampling them. We found this to be a good strategy when training in a low uncertainty setting, i.e., when most particles remain close to the underlying true states. Soft-resampling worked better under high uncertainty, where most particles would deviate far from the true states.

We have now introduced the PF-net architecture in a general setting. When applying PF-net to a particular task, we must choose the representation of states, and the network architecture for T and Z. Note that we may use different number of particles during training and during evaluation.

4 Visual localization

We apply PF-net to visual localization (Figure 1). A robot navigates in an indoor environment it has not seen before. The robot is uncertain of its location. It has an onboard camera, odometry, and it receives a schematic 2-D floor map. The task is to periodically estimate the location from the history of sensor observations. Formally, we seek to minimize the mean squared error,

$$\mathcal{L} = \sum_t (\overline{x}_t - x_t^*)^2 + (\overline{y}_t - y_t^*)^2 + \beta(\overline{\phi}_t - \phi_t^*)^2, \tag{9}$$

where $\overline{x}_t, \overline{y}_t, \overline{\phi}_t$ and x_t^*, y_t^*, ϕ_t^* are the estimated and true robot poses for time t, respectively; β is a constant parameter.

Challenges are threefold. First, we must periodically update a posterior over states given ambiguous observations, where the posterior is a multimodal, non-Gaussian, continuous distribution. PF-net tackles the challenge by encoding suitable differentiable algorithm prior, i.e., particle filtering.

Second, we need to build an observation model, $Z(o_t|s_t^k; \mathbb{M})$, that defines the probability of a camera observation, o_t, conditioned on the particle state, s_t^k, and the floor map, $\mathbb{M}$. A generative observation model would be hard to learn: it is a conditional distribution of images that show environments with different layouts, different furniture in different configurations, etc. PF-net learns a discriminative model instead, which takes s_t^k, o_t and $\mathbb{M}$ as inputs, and the particle likelihood as output. The discriminative model only needs to learn features relevant to localization. Importantly, PF-net learns the discriminative model end-to-end, without supervision on the particle likelihoods.

Third, the observation model must compare geometry extracted from a schematic 2-D map and a camera image. This is especially hard for visually rich environments, where some objects in the environment are not in the map, e.g., furniture. We introduce a neural network with appropriate structure for the observation model (Figure 3). The observation model defines the particle likelihood by appropriately combining features of the map and the camera image. First, a *local map* is obtained from $\mathbb{M}$ and s_t^k through an affine image transformation. For a neural network implementation we adopt the Spatial Transformer Network [39]. An affine transformation matrix, A^k, is computed for each particle, such that the transformed map is a local view from the pose, s_t^k. Because of this transformation, our network applies to any map size. Next, we extract features both from the local map and the camera image through separate learned CNN components. The feature maps are concatenated, fed to a locally fully connected layer, reshaped to a vector, and fed to a final fully connected layer. The output is a scalar, f_t^k, the likelihood of the particle state.

The details of the PF-net are as follows. Inputs are image observations, o_t, odometry, u_t, and the map, $\mathbb{M}$. The output is the continuous pose, $\overline{s_t} = \{\overline{x_t}, \overline{y_t}, \overline{\phi_t}\}$. Particles are pairs of a candidate pose, s_t^k, and weight, w_t^k. The output is the weighted mean of particles. The loss for end-to-end training, $\mathcal{L}$, is defined by (9). The observation and transition models, Z and T, are neural network components. We discussed the observation model above. The transition model updates the particle state given the odometry input, u_t, which is the relative motion defined in the robot coordinate frame. Our neural network transforms u_t to the global coordinate frame, u_t', and adds it to the previous pose along with Gaussian noise. Formally, we sample the new particle state from the differentiable function, $T(s_t^k|u_t', s_{t-1}^k) = s_{t-1}^k + u_t' + \mathrm{diag}(\sigma_t, \sigma_t, \sigma_r)\mathcal{N}(0; I)$, where $\mathcal{N}(0; I)$ is noise input from a standard multivariate Gaussian; σ_t and σ_r are standard deviations for translation and rotation, respectively. In experiments we chose σ_t and σ_r manually; however, they could be learned in the future.

5 Simulation experiments

We implemented[1] and evaluated PF-net in simulation for robot visual localization in indoor environments. We compared with several alternative methods to examine the benefits of differentiable algorithm priors and those of end-to-end training. We evaluated PF-net for various visual and depth sensor. Finally, we evaluated PF-net with increasing levels of uncertainty when the robot's initial belief changes from a distribution concentrated around its true pose to that of one spread uniformly over the entire space. The results are summarized in Table 1.

5.1 Experimental setup

Simulation. We conducted experiments in the House3D simulator [22], which builds on a large collection of human-designed, realistic residential buildings from the SUNCG data set [41]. On average, the building size is $206\,\mathrm{m}^2$, and the room size is $37\,\mathrm{m}^2$. See Figure 4 for examples.

Tasks. We consider localization with various levels of uncertainty. For *tracking*, the initial belief is concentrated around the true state. For *global localization*, the belief is uniform over all rooms in a building. In between, for *semi-global localization*, the belief is uniform over one or more rooms.

[1] Our Tensorflow implementation is available at `https://github.com/AdaCompNUS/pfnet`.

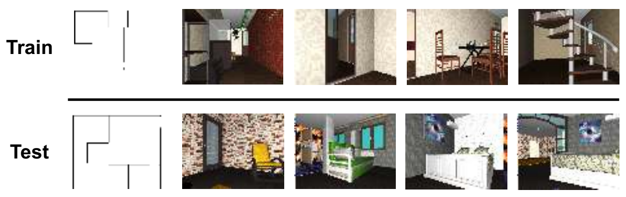

Figure 4: Sample training and test data from the House3D data set.

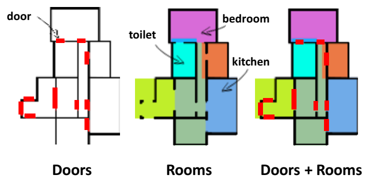

Figure 5: Semantic maps labeled with doors and room types.

Sensors. We considered a monocular RGB camera, a depth camera, an RGB-D camera, and a simulated 2-D LIDAR. Following earlier work [35], our simulated LIDAR simply transforms a depth images to a 2-D laser scan. The simulated LIDAR has a limited resolution of 54 beams and $60°$ field of view. As a result, localization with the simulated LIDAR is harder compared with a typical real-world LIDAR. We also considered a simplified environment, LIDAR-W, for the LIDAR sensor by removing all furniture from the environment and leaving only the walls. This way, the corresponding floor map contains all geometric objects in the environment.

Training. The training data consists of 45,000 trajectories from 200 buildings. Trajectories are generated at random: the robot moves forward ($p = 0.8$) or turns ($p = 0.2$). The distance and the turning angle are sampled uniformly from the ranges $[20\,\text{cm}, 80\,\text{cm}]$ and $[15°, 60°]$, respectively. Each trajectory is 24 steps long, and each step is labeled with the true robot pose. The robot's initial belief b_0 is a multivariate Gaussian distribution. The center of b_0 is perturbed from the true pose according to a Gaussian with zero mean and covariance matrix $\Sigma = \text{diag}(30\,\text{cm}, 30\,\text{cm}, 30°)$, and the the covariance of b_0 is the same Σ. This setting corresponds to a tracking task. We trained PF-net and alternative networks to minimize the end-to-end loss (9). We trained by backpropagation through time, limited to 4 time steps. For training PF-net, we used $K = 30$ particles. We did not resample particles during training, as it is not required for short trajectories and concentrated initial beliefs.

Alternative methods. We compared PF-net with alternative network architectures, histogram filter (HF) network [16] and LSTM network [36]. HF network represents the belief as a histogram over discretized states, in this case, a grid with $40\,\text{cm} \times 40\,\text{cm}$ cells and 16 orientations. Finer discretization did not produce better results. The LSTM network relies on its hidden state vector to represent the belief. We used a network architecture based on local maps, similar to the PF-net observation model. Outputs are relative state estimates that are updated with the odometry. We also considered a conventional particle filtering (PF) method with a handcrafted analytic observation model. We used the beam model implementation from the AMCL package of ROS [42], a standard model for localization with LIDAR [31]. The model parameters were tuned for our simulated LIDAR sensor. Finally, to calibrate the results, we also considered Odometry-NF, which updates the belief only with odometry, not with other sensor inputs.

Evaluation. We evaluated the methods on a fixed set of 820 trajectories in 47 previously unseen buildings for tracking, semi-global localization, and global localization tasks. We used the same setup and same model and algorithm parameters for all methods whenever possible. We trained the networks once for the tracking task and did not retrain for the other tasks. It is important to observe that for PF-net, the number of particles, K, used in execution does not have to be the same as that for training. In particular, we used $K = 300$ particles for tracking, $K = 1,000$ for semi-global localization, and K up to $3,000$ for global localization. We also activated resampling for semi-global and global localization. The same settings were applied to the PF method. For tracking, we report the average root mean squared error (RMSE), computed for the robot position (Table 1a). For semi-global and global localization, we report success rate on 100-step long trajectories (Table 1b–c). Localization is successful if the estimation error is below $1\,\text{m}$ for the last 25 steps of a trajectory. Finally, we evaluated PF-net on semi-global localization with semantic maps (Table 1d).

5.2 Main results

PF-net successfully reduces state uncertainty in the tracking task (Table 1a). Without additional training, PF-net can also localize successfully when the initial belief is uniform over a room (Table 1b), and even when uniform over the entire floor map (Table 1c). See Figure 6 for an example.

	RGB	Depth	RGB-D	LIDAR	LIDAR-W
PF-net	**40.5**	**35.9**	**33.3**	**48.3**	31.5
HF network	92.0	91.6	89.8	95.6	92.4
LSTM network	66.9	58.8	60.3	74.2	64.4
PF	–	–	–	81.3	**31.3**
Odometry-NF	109.4	109.4	109.4	109.4	109.4

(a) RMSE (cm) for tracking.

K	$N=1$	$N=2$	All
500	80.0%	70.5%	46.1%
$1,000$	84.3%	80.1%	57.9%
$2,000$	87.3%	84.8%	68.5%
$3,000$	**89.0%**	**85.9%**	**76.3%**

(c) PF-net success rate (%) for semi-global and global localization, with K particles. The initial belief is uniform over $N=1$, $N=2$, or all rooms.

	RGB	Depth	RGB-D	LIDAR	LIDAR-W
PF-net	**82.6%**	**84.0%**	**84.3%**	**69.4%**	**86.6%**
HF network	2.7%	2.9%	4.5%	1.6%	2.2%
LSTM network	21.1%	24.4%	23.4%	17.2%	22.2%
PF	–	–	–	25.1%	86.2%
Odometry-NF	1.1%	1.1%	1.1%	1.1%	1.1%

(b) Success rate (%) for semi-global localization over one room.

Labels	RGB	Depth	RGB-D
None	82.6%	84.0%	84.3%
Doors	84.4%	83.9%	84.5%
Rooms	84.5%	86.2%	86.5%
Both	**84.6%**	**86.7%**	**87.2%**

(d) PF-net success rate (%) for semi-global localization with additional semantic information on doors, rooms types, or both.

Table 1: Experimental comparison on robot visual localization.

Differentiable algorithm priors are useful. PF-net consistently outperformed alternative end-to-end learning architectures, HF network and LSTM network. Why? PF-net encodes differentiable algorithm prior specific to sequential state estimation, i.e., the particle representation of beliefs and their Bayesian update. HF network encodes similar prior for updating beliefs, however, it is restricted to a discrete belief representation which does not scale well to large and continuous state spaces. The LSTM network is not restricted to a discrete state space, but it has no structure specific to probabilistic state estimation, and it must rely on the hidden state vector to encode the belief.

End-to-end learning leads to increased robustness. We compared learned PF-net to PF with a known LIDAR model (first and fourth row of Table 1a–b). PF-net and PF performed similarly when only walls were present in the environment (*LIDAR-W* column). PF-net performed significantly better when some objects in the environment were not in the map (*LIDAR* column). Why? The beam model has no principled way to distinguish relevant walls from irrelevant objects, because it decouples the LIDAR scan to individual beams. Through end-to-end training, PF-net may have learned relationships between beams to distinguish walls from objects. PF-net may have also learned to deal with map imperfections, e.g., missing walls, glass doors, and various map artifacts, which we observed occasionally in the House3D data set.

PF-net is effective with various sensors. The columns of Table 1a-b compare different sensors for localization. PF-net with RGB images is almost as effective as with depth images; and it performs better than with simulated LIDAR. This indicates that PF-net successfully learned to extract relevant geometry from RGB images, and it learned to ignore objects that are not in the map. When combining RGB and depth image inputs, RGB-D column, performance improves. This demonstrates that PF-net can learn simple sensor fusion end-to-end from data. Future work in this direction is promising.

5.3 Additional experiments

Global localization. We evaluated learned PF-net for localization with increasing difficulty (Table 1c). We chose initial beliefs uniform over one room, two rooms, and the entire building. We compared PF-net with different number of particles, up to $K = 3000$. Results show that PF-net can solve global localization with high initial uncertainty when provided with sufficiently many particles.

Semantic maps. Humans often use floor maps with semantic information: there are labels for the office, toilet, lift and staircase. Utilizing semantic maps for robot localization is not trivial [35, 43]. PF-net may learn to use semantic maps naturally, through end-to-end training. To demonstrate this, we trained PF-nets with simplified semantic maps with labels for doors and room categories. See Figure 5 for examples. We encoded the semantic labels in separate channels of the input map: one

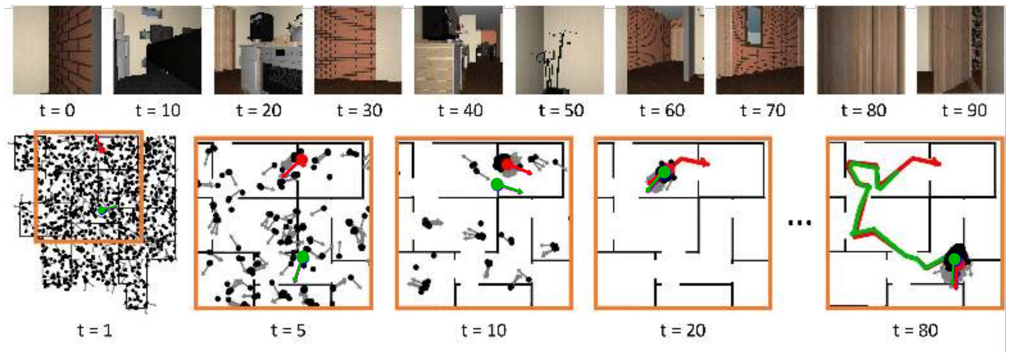

Figure 6: Global localization with PF-net ($K = 1000$) and RGB camera input. Red indicates ground-truth poses. Black indicates PF-net particles. Green indicates the weighted mean of particles.

channel for doors, 8 channels for 8 distinct room categories. Results show that simple semantic maps can indeed improve localization performance (Table 1d).

Ablation study. In supplementary experiments we altered certain settings of PF-net during training, and evaluated the learned PF-nets for a fixed semi-global localization task. First, we added soft-resampling during training. When trained for the tracking task as before, success rates decreased for soft-resampling: 79% to 75%. However, when trained with increased initial uncertainty and noisy odometry, success rates increased for soft-resampling: 39% to 42%. As expected, resampling can be beneficial when most particles would be far from the true state; but it hurts when particles near the true state are eliminated, which often happens in early phases of learning. Future work may incorporate various strategies for resampling only when required [44]. Indeed, when resampling only every second step, success rates increased: 42% to 54%.

Next, we varied the number of backpropagation steps for BPTT. Backpropagating through multiple steps improved performance: 73%, 79%, 79% success rates for 1, 2, and 4 steps, respectively. This indicates that loss from future steps can provide a useful learning signal for the present step.

Finally, we replaced our loss function (9), with the probabilistic loss function proposed in [26]. The alternative loss function worked worse when training in the standard tracking setting, 74% versus 79% success rates. However, the alternative loss function worked better when training with increased uncertainty, 67% versus 39%. Our loss can be dominated by the distant particles, which may negatively affect learning in the latter case.

6 Conclusion & future work

We introduced the PF-net, a neural network architecture with differentiable algorithm prior for sequential state estimation. PF-net encodes learned probabilistic models, together with a particle filter algorithm, in a differentiable network representation. We applied PF-net to robot localization on a map. Through end-to-end training, PF-net successfully learned to localize in challenging, previously unseen environments populated with objects not shown in the map.

Future work may apply PF-net to real-world localization, a problem of great interest for mobile robot applications. One concern is online execution. With RGB input PF-net needs approx. 0.6ms per particle per step. Indoor localization with high uncertainty may require up to 1,000 – 10,000 particles [8]. We can increase robustness, and use less particles, by incorporating standard techniques for particle filtering, e.g., injecting particles and adaptive resampling [44]. We may also improve inference time, leveraging an abundance of work optimizing neural network models and hardware [45]. Finally, learned PF-net models can be used for standard particle filtering, and thus visual sensors can be complementary to laser, potentially at a lower update frequency.

PF-net could also be applied to other domains, e.g., visual object tracking and SLAM. An exciting line of future work may extend PF-net to learn latent state representations for filtering, potentially in an unsupervised setting. Finally, the particle representation of beliefs can be important for encoding more sophisticated algorithms in neural networks, e.g., for planning under partial observability.

Acknowledgments

This research is supported in part by Singapore Ministry of Education grant MOE2016-T2-2-068. Peter Karkus is supported by the NUS Graduate School for Integrative Sciences and Engineering Scholarship.

References

[1] A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[2] S. Thrun. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002.

[3] A. Blake and M. Isard. The condensation algorithm-conditional density propagation and applications to visual tracking. In *Advances in Neural Information Processing Systems*, pages 361–367, 1997.

[4] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003.

[5] P. Del Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[6] J. S. Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[7] A. Doucet, N. De Freitas, and N. Gordon. Sequential monte carlo methods in practice. series statistics for engineering and information science, 2001.

[8] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1-2):99–141, 2001.

[9] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598, 2002.

[10] N. Ye, A. Somani, D. Hsu, and W. S. Lee. Despot: Online POMDP planning with regularization. *Journal of Artificial Intelligence Research*, 58:231–266, 2017.

[11] G. Shani, R. I. Brafman, and S. E. Shimony. Model-based online learning of POMDPs. In *European Conference on Machine Learning*, pages 353–364, 2005.

[12] B. Boots, S. M. Siddiqi, and G. J. Gordon. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7):954–966, 2011.

[13] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3(Dec):679–707, 2002.

[14] A. Tamar, S. Levine, P. Abbeel, Y. Wu, and G. Thomas. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2146–2154, 2016.

[15] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel. Backprop kf: Learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems*, pages 4376–4384, 2016.

[16] R. Jonschkowski and O. Brock. End-to-end learnable histogram filters. In *Workshop on Deep Learning for Action and Interaction at NIPS*, 2016.

[17] P. Karkus, D. Hsu, and W. S. Lee. QMDP-net: Deep learning for planning under partial observability. In *Advances in Neural Information Processing Systems*, pages 4697–4707, 2017.

[18] J. Oh, S. Singh, and H. Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, pages 6120–6130, 2017.

[19] G. Farquhar, T. Rocktäschel, M. Igl, and S. Whiteson. TreeQN and ATreeC: Differentiable tree planning for deep reinforcement learning. *arXiv preprint arXiv:1710.11417*, 2017.

[20] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145, 2017.

[21] T. Shankar, S. K. Dwivedy, and P. Guha. Reinforcement learning via recurrent convolutional neural networks. In *International Conference on Pattern Recognition*, pages 2592–2597, 2016.

[22] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

[23] A. Guez, T. Weber, I. Antonoglou, K. Simonyan, O. Vinyals, D. Wierstra, R. Munos, and D. Silver. Learning to search with MCTSnets. *arXiv preprint arXiv:1802.04697*, 2018.

[24] M. Okada, L. Rigazio, and T. Aoshima. Path integral networks: End-to-end differentiable optimal control. *arXiv preprint arXiv:1706.09597*, 2017.

[25] P. Donti, B. Amos, and J. Z. Kolter. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 5484–5494, 2017.

[26] R. Jonschkowski, D. Rastogi, and O. Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv preprint arXiv:1805.11122*, 2018.

[27] C. A. Naesseth, S. W. Linderman, R. Ranganath, and D. M. Blei. Variational sequential monte carlo. *arXiv preprint arXiv:1705.11140*, 2017.

[28] C. J. Maddison, J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pages 6576–6586, 2017.

[29] T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. Auto-encoding sequential monte carlo. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[30] S. Gu, Z. Ghahramani, and R. E. Turner. Neural adaptive sequential monte carlo. In *Advances in Neural Information Processing Systems*, pages 2629–2637, 2015.

[31] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[32] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 588–594. IEEE, 1999.

[33] P. Elinas and J. J. Little. σmcl: Monte-carlo localization for mobile robots with stereo vision. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[34] B. Coltin and M. Veloso. Multi-observation sensor resetting localization with ambiguous landmarks. *Autonomous Robots*, 35(2-3):221–237, 2013.

[35] O. Mendez, S. Hadfield, N. Pugeault, and R. Bowden. Sedar-semantic detection and ranging: Humans can localise without lidar, can robots? *arXiv preprint arXiv:1709.01500*, 2017.

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[37] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[38] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[39] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[41] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 190–198, 2017.

[42] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, 2009.

[43] O. Mozos and W. Burgard. Supervised learning of topological maps using semantic information extracted from range data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[44] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[45] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.